所属类别	2021 年“华数杯”全国大学生数学建模竞赛	参赛编号

基于机器学习的电动汽车目标客户销售策略研究

摘要

本文探究的是某汽车公司中三个品牌电动汽车的目标客户满意度情况、影响因素、预测购买可能性以及制定销售策略等问题。

针对问题一，通过分析附件一，同时结合附件二的个人特征调查表，提取调查表问题本身的限制条件，并发掘调查问题之间的关系，最终建立不同调查问题之间合理的约束关系，由此，对不同的异常数据和缺失数据采用了不同的填充策略。基于数据清洗后的结果，对各项影响满意度得分的指标取平均值，**发现**：目标客户对该公司三种品牌的满意度从高到低依次为合资品牌>自主品牌>新势力品牌。

针对问题二，采用了基于惩罚项和基于树模型两种不同的嵌入法分别进行特征选取，前一种用 LR、LASSO、SVM 三种模型，后一种用 RF、LightGBM 模型共五个模型来进行机器学习，求出影响不同品牌销售的相关特征，对选出的特征上采用投票法，**结果发现**：电动车的电池技术性能、舒适性、目标客户全年房贷、车贷占家庭年收入情况对三种品牌的销售均有较大影响。除此之外，经济型、安全性、客户的工作情况等也在不同程度上对不同品牌的销售产生不同程度的影响。

针对问题三，由于标签不均衡的问题严重，本文先通过 SMOTE 采样解决标签不平衡问题，再通过纵向比较和横向比较，纵向使用 F1-score 和 AUC 指标衡量训练效果，横向对不同模型的分数进行比较，从而挑选出适用于不同品牌的最优模型。进而利用网络搜索方法给每个品牌效果最优的模型进行超参数调优，其中 LightGBM 的在整个验证集上的 AUC 值达到 97.10%。最后利用这三个最优模型预测了附件三的 15 名目标客户购买电动车的可能性。

针对问题四，本文在问题三基础上，运用了多目标规划原理，建立三个目标函数，即提高体验满意度的服务难度尽量小，选择提高服务的数量尽可能少，目标客户购买概率提升的百分比尽可能大。用 python 求解该规划模型，**结果显示**，着重满足品牌 1、编号 3 的顾客在经济性上的需求；满足品牌 2、编号 9 的顾客在电池性能上的需求以及品牌 3、编号 14 的顾客在舒适性上的需求有助于提高顾客的购买率。

针对问题五，基于问题一到问题四所建立的模型、因子探究等的基础上，提出了精准营销的策略，在人群洞察、品牌导向、效果评估这三个方向去探讨，旨在为该团队有针对性地制定销售策略，为该汽车公司创造最大的效益。

关键词：销售策略　特征工程　LightGBM　多目标规划　随机森林

一、 问题重述

1.1 问题背景

在近年来我国提出早日实现“碳达峰”、“碳中和”目标的背景下，大力发展新能源电动汽车被认为是解决能源环境问题的一种有效途径，具有广阔的市场前景。习近平总书记在 2014 年考察上海汽车集团时也曾强调“发展新能源汽车是迈向汽车强国的必由之路”[1]。但与传统汽车相比，百姓对新能源汽车的电池仍存在一定的疑虑，因此市场销售需要科学决策，针对目标客户制定出对应的营销策略，从而增加购买率，为公司创造收益。

1.2 问题提出

某汽车公司最新推出了三款品牌的电动汽车，分别是合资品牌、自主品牌和新势力品牌。为了研究消费者对该公司电动汽车的购买意愿，并制定相应的销售策略以科学决策，销售部门邀请了 1964 位目标客户对三款品牌的电动汽车进行体验。现要求利用数学模型的知识解决下列问题：

1. 请做数据清洗工作，指出异常值、缺失数据及处理方法，对数据做描述性统计分析，比较分析目标客户对不同品牌汽车的满意度。
2. 根据目标客户在体验活动中购买电动车的情况，探究可能会影响不同品牌电动汽车销售的因素。
3. 结合前面的研究结果，建立不同品牌电动汽车的客户挖掘模型，评价模型的优良性，并判断给定目标客户购买电动汽车的可能性。
4. 虽然营销者加大服务力度可在短时间内提高满意度，但服务难度与满意度的提高成正比。基于这个思路及研究成果，在三个品牌中各挑选一名没有购买电动汽车的目标客户，实施营销策略。
5. 根据研究结论，为营销部门提出不超过 500 字的营销策略建议。

二、 问题分析

2.1 问题一的分析

问题一要求考虑题目要求和结合实际背景对附件一和附件三的数据进行数据清洗操作，并进行描述性统计，该题的思路分为一下三个步骤：

1. 利用附件二的目标客户个人特征调查表，以及题目中对满意度评价表的约束，综合考虑实际生活和客观规律，充分挖掘问题本身的限制条件，以及不同问题之间的限制条件，最终建立不同回答之间合理的约束关系。

2. 之后利用所建立出的约束关系，对附件一和附件三的异常数据进行初步清洗。对于离群类型异常值用有效数据的平均值替换。对于不满足约束关系的异常值，用数据间的关系做出推断后进行替换，对缺失数据的缺失情况进行综合分析，用已有的数据进行推断并分析实际背景，对缺失数据进行填充处理，以保证建模的准确性。
3. 基于数据清洗后的结果，对各项影响满意度得分的指标取平均值，然后对各品牌进行排序。

2.2 问题二的分析

问题二需要筛选出影响不同品牌销售的主要因素，针对该问题采用基于惩罚项和基于树模型两种不同的嵌入法，前一种用 LR、LASSO、SVM 三种模型，后一种用 RF、LightGBM 模型共五个模型，求出影响不同品牌销售的相关特征。再根据特征重要性程度从高到低排序，并设置一个阈值，得到每个算法重要性排名靠前的特征，在五个模型选出的结果上采用投票法，筛选出大于等于 2 票的特征，作为影响该品牌销售主要因素。

2.3 问题三的分析

问题三是建立在问题二的基础上的，通过 SMOTE 采样解决标签不平衡问题之后，利用 F1-score 和 AUC 指标，对模型的训练和预测效果进行纵向比较，并利用 k 折交叉验证，对不同模型进行横向比较。对三个品牌，分别训练这 5 个模型，通过比较分析，在不同品牌中，挑选出在验证集上 F1-score 和 AUC 值最优的模型，并利用网络搜索方法，给每个品牌对应的最优的模型进行超参数调优。最后利用这三个最优模型预测附件三的 15 名目标客户购买电动车的可能性。

2.4 问题四的分析

问题四，是在问题二、三基础上的多目标规划问题。需要建立以提高服务的难度最小，提高的服务的数量最少，购买概率提高的百分比最多的多目标规划模型。在附件三的 3 个品牌中挑选没有购买电动车的客户，求解得出使得各个体验满意度应该提高的百分比，对没有购买电动车的目标客户设计销售策略。

2.5 问题五的分析

问题五主要是建立在前四问的基础上以信件的方式对销售部门提出销售策略。根据问题一可以得出客户对不同品牌的倾向程度；根据问题二可以得出对三种品牌电动汽车销售影响较大的因子；根据问题三可以得出适合三种品牌预测目标客户购买可能性的模型；根据问题四可以在提升服务满意度与服务难度之间找到一个优化方案。

三、 模型假设

1. 假设影响目标客户满意度的因素仅考虑题目给出的电池技术性能、舒适性、经济型、安全性、动力性、驾驶操控性、外观内饰整体表现和配置与质量品质，且各个影响因素之间相互独立。
2. 假设服务难度与提高的满意度百分点是成正比的关系。
3. 假设在做出销售策略的这段时间内，目标客户对电动汽车的满意度、购买意向等不会发生改变。
4. 假设在做出销售策略的阶段不会有黑天鹅事件或重大自然灾害等对该公司的电动车销售带来影响。

四、 符号说明

符号	说明	单位
$\hat{y}$	制定策略后的购买概率	%
y	原始购买概率	%
c_i	提高服务项目（提高为 1，否则为 0）	1
$\hat{x_i}$	第 i 个指标提高后的结果	1
w	叶子节点的分数	1
k	树的数量	棵
T	叶子节点的个数	个

五、 模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 数据预处理

缺失值、异常值的处理是数据处理不可缺少的一环。缺失值是源于数据采集的空缺、传输间丢失等不可控情况所导致或人为故意丢失等多种情况，如何处理空值是数据处理中恒久不变的问题。而其中异常值是由于数据传输错误所导致的，通常对其采取修改或是剔除的处理方式，但具体所采取的方式也需要依据客观上分析数据所决定。

以下主要分成三点叙述本文所采用的处理方法及处理结果。

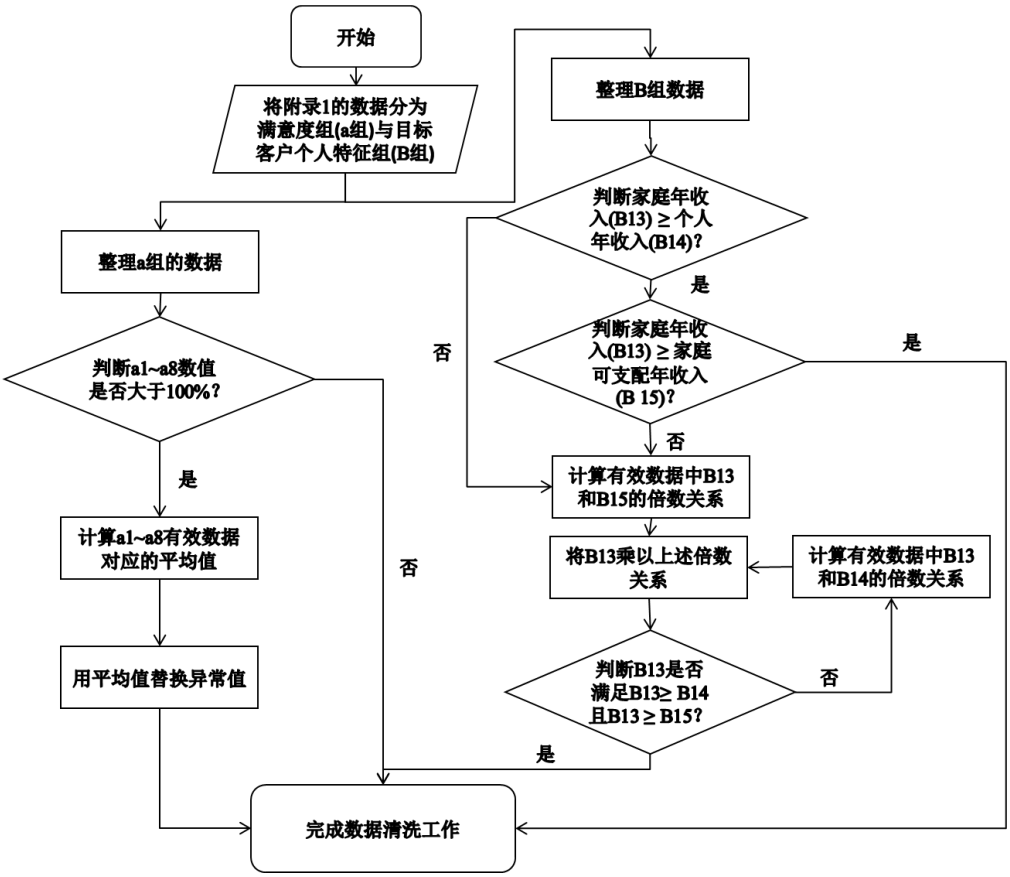

图 1. 异常数据清洗的整体思路

（一） 满意度得分的异常数据清洗工作

根据题意，该公司的销售部门对目标客户进行体验调查时，规定满意度得分满分为 100 分，故在数据清洗时剔除附录 1 中 a1~a8 数值大于 100 的数据。以 a1 为例，该列展示了目标客户对电池技术性能的满意度得分情况。利用散点图经筛查发现，编号为 0001 的目标客户对电池技术性能的满意度高达 753.04 分（> 100 分），故用正常值数据的平均值（77.93 分）进行替换。数据清洗前后的结果如图 2、图 3 所示。同理可分别对编号为 1964、0480 的目标客户对应在 a3、a5 的数据进行清洗（满意度原始评价分数分别为 703.00 分、605.03 分）。

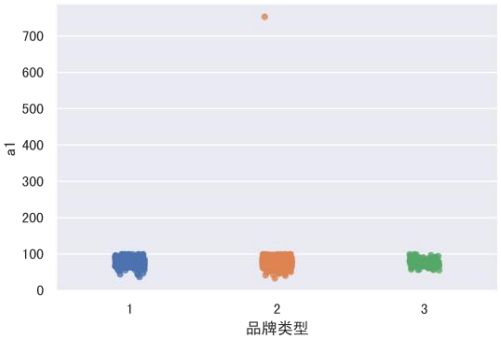

图 2. 附录 1 中 a1 的原始数据散点图

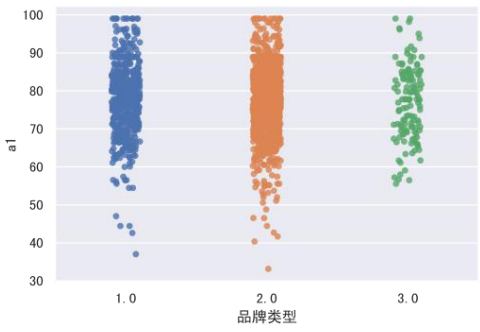

图 3. 附录 1 中 a1 清洗后的数据散点图

（二） 目标客户个人特征的异常数据清洗工作

1. 由于家庭年收入应大于等于个人年收入，故对附录 1 中 b13、b14 的数据进行筛查，共发现 76 条异常值。同理，家庭年收入应大于等于家庭的可支配年收入，故对附录 1 中 b13、b15 的数据进行筛查，共发现 73 条异常值。

考虑到不同目标客户的家庭、个人收入极值差异较大，为了提高替换数据的准确度，故先以家庭可支配收入为基准，计算得有效数据中各个家庭的可支配年收入占年收入的比例，并对所有家庭该比例值进行排序，求得中位数为 0.6667，再用这个倍数关系，修正家庭的年收入值，并对修正后的年收入值与个人年收入进行比较、校验。

清洗后的数据经检验仍有 7 条数据不满足上述规则，故再用相同的方法，以个人年收入为基准，计算得有效数据中个人年收入占家庭年收入比例的中位数为 0.7%，进行数据清洗后按照规则进行校验，满足规则要求。

2.由附录 2 的目标客户个人特征调研表可知，B17 调查了目标客户全年车贷支出占家庭年收入的比例，故用正常值的平均值（10%）替换附录 1 中该题大于 100%的数值，即编号为 0223 的客户，其 B17 问题的原始数值高达 300%，数据清洗前后的散点图如图 4、图 5 所示。

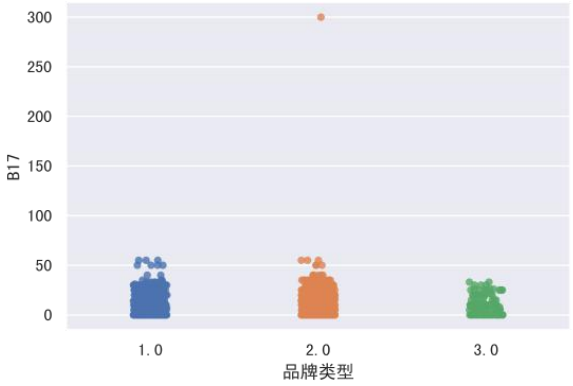

图 4. 附录 1 中 B17 的原始数据散点图

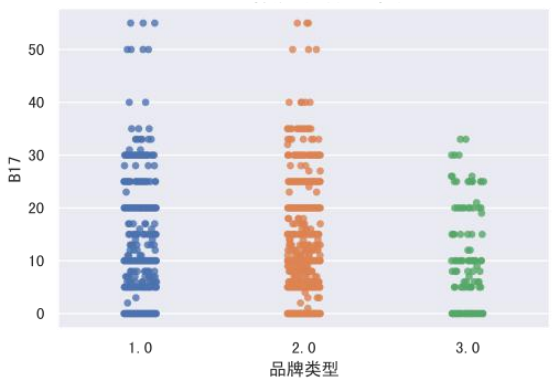

图 5. 附录 1 中 B17 清洗后的数据散点图

（三） 缺失值的处理

由附录 1 可知，缺失数据集中在 B7，即目标客户的孩子数量。由于目前已有的数据仅为 1、2、3 名孩子数量，不符合常理，因此考虑了出生年份后，将所有缺失数据的孩子数量均替换为 0，共完成 1457 处的替换。

5.1.2 目标客户对不同品牌汽车的满意度分析

根据题意可知，影响目标客户对三款品牌电动汽车满意度的主要影响因素主要有 8 个，分别为电池技术性能（电池耐用和充电方便）、舒适性（环保与空间座椅）整体表现、经济性（耗能与保值率）、安全性表现（刹车和行车视野）、动力性表现（爬坡和加速）、驾驶操控性表现（转弯和高速的稳定性）、外观内饰整体表现、配置与质量品质，其满意度分别是 a1~a8。为此，基于上述数据清洗的结果，对全体目标客户评价的 a1~a8 分别求平均值，然后再对这 8 项指标的平均值再次求平均，并求出最后得

分。运行结果显示，合资品牌、自主品牌和新势力品牌的满意度最终得分分别为 78.16 分、77.54 分和 76.93 分，表明目标客户普遍对合资品牌的满意度评价最高，新势力品牌的满意度最低。

5.2 问题二模型的建立与求解

特征选择方法中的过滤法和包裹法，在特征选择过程与模型训练过程是独立进行的，而嵌入法则是综合考虑这两个过程，在学习的同时进行特征选择，因此，为了使结果更具有代表性，利于模型训练，本文使用以下两种嵌入法：

一是基于惩罚项的特征选择法，本文采用基于 SVM 模型、LASSO 模型、逻辑回归模型的惩罚项特征选择法来选择特征。用正则 L1 范数作为惩罚项。L1 范数不但 可以降低过拟合风险，还可以使求得的 w 有较多分量为 0。所以当希望减少特征 的维度以用于其他分类器时，可以选择不为 0 的系数所对应的特征

二是树模型的特征选择法，这种方法能够用来计算特征的重要程度，因此可以用来去除不相关的特征。本文采用随机森林模型、LightGBM 模型的树模型特征选择法来选择特征。再根据特征重要性程度，设置一个阈值，得到每个算法排名靠前的特征，

最后在五个模型选出的结果上采用投票法，筛选出大于等于 2 票的特征，作为影响该品牌销售主要因素。

5.2.1 逻辑回归模型的建立

逻辑回归（logistic regression，LR）是一个强大的统计学方法，它可以用一个或多个解释变量来表示一个二项式结果[2-5]。它通过使用逻辑函数来估计概率，从而衡量类 别依赖变量和一个或多个独立变量之间的关系，后者服从累计逻辑分布。它的定义式如下：

$$h(z) = \frac{1}{1 + exp\,(-z)}$$

这个函数的定义域为整个实数域，值域为（0，1），并且是一个单调的增函数。根据对分布函数的要求，这个函数可以用来作为随机变量 x 的分布函数，即：

$$p(x \le z) = h(z) \tag{1}$$

5.2.2 LASSO 回归模型的建立

LASSO 的全称是 Least absolute shrinkage and selection operator，是一种压缩估计，它通过构造一个惩罚函数得到一个较为精炼的模型，使得它压缩一些回归系数，即强制系数绝对值之和小于某个固定值，同时设定一些回归系数为零[6,7]。因此保留了子集收缩的优点，是一种处理具有复共线性数据的有偏估计。LASSO 是在 RSS 最小化的计算中加入一个 l_p 范数作为惩罚约束。l_p 范数的好处是当λ充分大时，可以把某些待估系数精确地收缩到零。通过交叉验证法：对λ的给定值，进行交叉验证，选取交叉验证误差最小的λ值，然后按照得到的λ值，用全部数据重新拟合模型即可。

$$\widehat{\beta_{Lasso}} = arg\min_{\beta \epsilon R^d}(\|Y - X\beta\|^2 + \lambda\sum_{j=1}^{d}|\beta_j|) \quad (1)$$

其中 t 与λ一一对应，为调节系数。

5.2.3 SVM 模型的建立

SVM 支持向量机主要针对二分类线性分类器，方法是确定一个超平面，使得数据点到超平面的距离最大，这是一个二次规划问题[8-10]。由于对任何数据集找到它合适的映射是困难的，因此通常会从常用核函数中选择[11]。常用的核函数有多项式核函数、高斯函数、线性函数等。分类超平面的约束为：

$$y_i(w^T x_i + b) \geq 1 \quad (1)$$

目标函数是超平面离两类样本的距离要足够大，可以写成：

$$\frac{1}{2}||w||^2 \quad (2)$$

如果再加上松弛变量ξ_i和惩罚因子 C 对违反不等式的样本进行惩罚，可以得到如下最优化问题：

$$\begin{aligned} &min\frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \\ &y_i(w^T x_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0, i = 1,2,\cdots,l \end{aligned} \quad (3)$$

5.2.4 随机森林模型的建立

随机森林是指利用多棵决策树对样本进行训练并预测的一种算法[12,13]。随机森林算法是一个包含多个决策树的算法，其输出的类别是由个别决策树输出类别的种树来决定的。它的优点在于大部分的数据，它的分类效果比较好，能处理高维特征，不易产生过度拟合，大数据的模型训练速度较快，在决定类别时可评估变数的重要性，且对数据集的适应能力强，既能处理离散型数据也能处理连续型数据，数据集无需特意规范化。随机森林是基于 Bagging 思想的集成学习模型，通过构建多个学习器共同完成学习任务[14,15]。该模型 Bagging 算法原理类似投票，每次使用一个训练集训练一个弱学习器，有放回地随机抽取 n 次后，根据不同的训练集训练出 n 个弱学习器。对于分类问题，根据所有的弱学习器的投票，进行“少数服从多数”的原则进行最终预测结果。对于回归问题，采取所有学习器的平均值作为最终结果。在决策树中采用基尼系数的统计增益。基尼系数表示数据集中样本的差异程度，基尼系数越大，表示数据集的种类越多样，即表示有多种的分类结果，表示数据集当前特征的越多样，即越不纯，即可能是一个多分类的问题。

具体公式如下：

$$Gini(D) = 1 - \sum_{k=1}^{|y|} p_k^2 \tag{1}$$

一般选择基尼系数最小的特征作为最初的根节点。在实际项目中根据定义的分类结果进行根节点的选择，通过统计训练样本的特征与类别，根据特征相对初始类别的基尼系数的增益决定之后每一步根节点的选择。在计算基尼系数增益之前需要知道当对每个特征作为判断依据时的基尼指数，具体公式如下：

$$Gini_{index}(D, a) = \sum_{v=1}^{v} \frac{|D^v|}{|D|} Gini(D^v) \tag{2}$$

基尼增益为：

$$Gini(D, a) = Gini(D) - Gini\_index(D, a) \tag{3}$$

本文通过运行程序不断对内部节点进行分类直至显示最终的分类结果为止。此时对样本训练完毕，当有新的样本来进行测试时我们根据当前模型进行预测。在实际工程中需要控制决策树的深度防止过拟合的情形，即在训练样本可以表现很好的性能，但是并不能保证测试样本的准确度，这在训练集很大的时候容易出现。

5.2.5 LightGBM 模型的建立

LightGBM 是一款基于决策树算法的分布式梯度提升框架。它的优点在于减少了数据对内存的使用，保证单个及其在不牺牲速度的情况下尽可能使用多的数据，同时减少通信的代价，提升多级并行时的效率，实现在计算上的线性加速[12,16,17]。

5.2.6 基于树模型特征法的选择特征结果

本文采用随机森林模型、LightGBM 模型的树模型特征选择法来选择特征，计算结果如表 1~2 所示。

表 1. 基于惩罚项的嵌入法选取的特征

特征选取方法	选择结果		
	品牌 1	品牌 2	品牌 3
LR 逻辑回归	a1 a2 a3 b4 b5 B15 B16 B17	a1 a2 a3 a4 a5 a7 B11 B15 B16 B17	a1 a2 a3 a5 a6 B15 B16
SVM 支持向量机	a1 a2 a3 b3 b6 B10 B16 B17	a1 a3 a5 a6 a8 b8 B10 B15 B16 B17	a1 a2 a3 a7 B2 B6 B16
LASSO 回归	a1 a2 a3 a6 b6 B12 B16 B17	a1 a3 a5 a6 a7 B10 B11 B13 B16 B17	a1 a2 a5 b6 B10 B16 B17

表 2. 基于树模型的嵌入法选取的特征

特征选取方法	选择结果		
	品牌 1	品牌 2	品牌 3
随机森林	a1 a2 a3 B2 B15 B16 B17	a1 a2 a3 a4 a5 a7 B13 B15 B16 B17	a1 a2 a5 a3 a6 a8 B16
LightGBM	a1 a4 a5 B6 B7 B10 B16 B17	a1 a3 a7 B1 B11 B13 B14 B15 B16 B17	a2 B16

图 6 是利用五种模型对各个特征进行投票得出的特征出现次数图，本文选取大于等于 2 出现次数的特征为重要特征，并认为会分别对三种品牌的电动汽车产生影响。

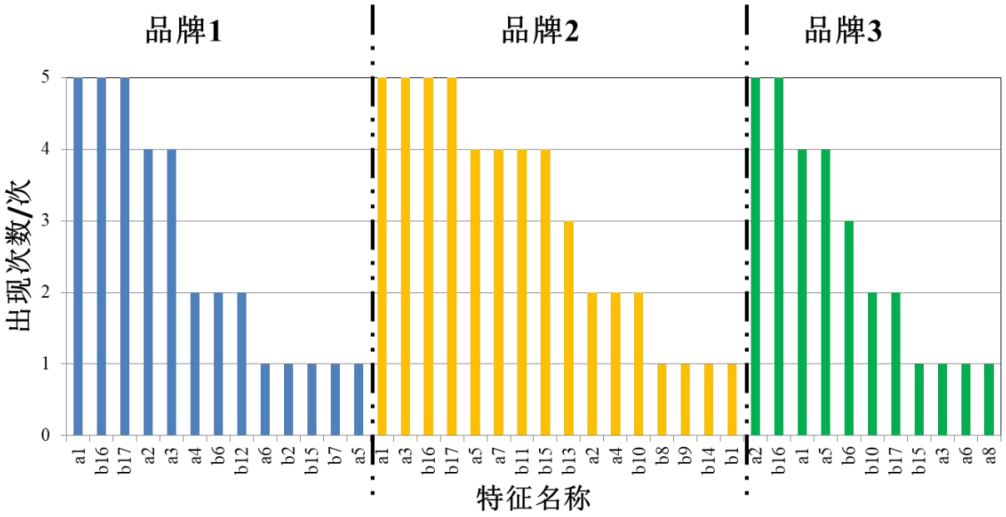

图 6. 特征出现次数图

由图 6 可知，对三种品牌来说，电动车的电池技术性能、舒适性、目标客户全年房贷、车贷占家庭年收入的占比均为两次及以上，说明这四个指标对顾客的购买情况有较大的影响。除了以上的四个指标外，影响品牌 1 购买情况的因子还有经济型、安全性、驾驶操控性以及目标客户的职位；经济型、安全性、动力性、外观内饰、目标客户的工作情况（包括工作年限、单位性质）、家庭年收入、可支配年收入会是目标客户购买品牌 2 的主要考虑因素；品牌 3 销售的主要影响指标则还有动力性、目标客户的婚姻家庭情况还有工作年限。

5.3 问题三模型的建立与求解

5.3.1 基础理论

对于类别不平衡问题，通过 SMOTE 进行采样，为了评价模型的优良性，我们使用 F1 值和 AUC 值对单个模型进行综合评价。同时对每个模型进行 k 折交叉验证，计

算 F1 值和 AUC 值，对不同模型的分数进行横向比较，从而挑选出适用于不同品牌的最优模型。进而利用网络搜索方法给每个品牌效果最优的模型进行超参数调优。

(一)F1 分数

F1 分数是用来衡量二分类模型精确度的一种指标。它同时兼顾了分类模型的准确率和召回率。F1 分数可以看作是模型准确率和召回率的一种加权平均，它的最大值是 1，最小值是 0，越接近于 1 表示模型越好[18-20]。

$$\frac{2}{F_1}=\frac{1}{P}+\frac{1}{R} \tag{1}$$

F 值可泛化为对精确率和召回率赋予不同权重进行加权调和：

$$F_a=\frac{(1+a^2)\times P\times R}{a^2\times P+R} \tag{2}$$

准确率和错误率也是常用的评估指标，精确率是一个二分类指标，准确率能应用于多分类：

$$\text{准确率}=\frac{TP+TN}{TP+FP+FN+TN} \tag{3}$$

$$\text{错误率}=\frac{FP+FN}{TP+FP+FN+TN} \tag{4}$$

$$\text{准确率}=\frac{1}{n}\sum_{i=1}^{n} I(f(x_i)=y_i) \tag{5}$$

准确率指的是预测值为 1 且真实值也为 1 的样本在预测值为 1 的所有样本中所占的比例。召回率指的是预测值为 1 且真实值也为 1 的样本在真实值为 1 的所有样本中所占的比例。

(二)AUC 值

AUC 是指 ROC 曲线下与坐标轴围成的面积，这个面积的数值小于等于 1。又因为 ROC 曲线一般都处于 y=x 这条直线的上方，所以 AUC 的取值范围在 0.5 和 1 之间。AUC 越接近 1.0，检测方法的实性越高；等于 0.5 时真实性最低，无应用价值。

$$AUC=\sum_{i\in(P+N)}\frac{(TPR_i+TPR_{i-1})\times(FPR_i-FPR_{i-1})}{2} \tag{6}$$

评价指标是针对同样的数据，输入不同的算法，或者输入相同的算法但参数不同而给出这个算法或者参数好坏的定量指标。本文将问题看成二分类问题，因此本文主要用到的评价指标有准确度、召回率、ROC 曲线以及 AUC 面积，这些是在二分类问题比较常见的指标。混淆矩阵是 ROC 曲线绘制的基础，同时它也是衡量分类型模型准确度中最基本、最直观、计算最简单的方法。以分类模型中的二分类为例，对于这种问题，模型最终需要判断样本的结果是 0 还是 1，或者说是 Positive 还是 Negative。因此，能得到这四个基础指标，可以看成是一级指标：真实值是 Positive，模型认为是 Positive 的数量(TP)，真实值是 Positive，模型认为是 Negative 的数量(FN)，真实值是 Negative，模型认为是 Positive 的数量(FP)，真实值是 Negative，模型认为是 Negative

的数量(TN)，将这四个指标一起呈现在表格中，就能得到如表 3 这样一个矩阵，称它为混淆矩阵。预测性分类模型，是希望越准越好。在混淆矩阵中，TP 与 TN 的数量越大越好，而 FP 与 FN 的数量越小越好。

表 3. 混淆矩阵

混淆矩阵		真实值	
		Positive	**Negative**
预测值	Positive	TP	FP
	Negative	FN	TN

(三)SMOTE 法

数据不平衡也可称作数据倾斜。在实际应用中，数据集的样本特别是分类问题上，不同标签的样本比例有大概率事件是不均衡的。因此，如果直接使用算法训练进行分类，训练效果可能会很差，本文主要是用 SMOTE 采样方法对数据进行采样以消除不平衡。

SMOTE 采样的原理是在少数类样本之间进行插值来产生额外的样本。具体地，对于一个少数类样本使用 k 近邻法(k 值需要提前指定)，求出离 x_i 距离最近的 k 个少数类样本，其中距离定义为样本之间 n 维特征空间的欧氏距离。然后从 k 个近邻点中随机选取一个，使用下列公式生成新样本：

$$x_{new} = x_i + (\hat{x_i} - x_i) \times \delta \tag{7}$$

其中，$\hat{x_i}$为选出的 k 近邻点，δ∈[0,1] 是一个随机数。

(四)K 折交叉验证

K 折交叉验证是将数据集 D 划分成 K 份互斥数据集 D_K，满足 $D=D_1 \cup \cdots \cup D_k$，一般是平均分配使每份数据量接近并且数据分布尽可能一致。每次用一份数据测试，其余 $K-1$ 份数据训练，需要迭代 K 轮得到 K 个模型；最后再将 K 份测试结果汇总到一起评估一个离线指标。

$$cv_{score} = \frac{1}{K}\sum_{k=1}^{K} L(P_k, Y_k) \tag{8}$$

K 折叉验证的稳定性与 K 取值有很大关系。K 值太小实验稳定性偏低，K 值太大又可能导致实验成本高，K 最常用的取值是 5 和 10。K 折交叉验证能够更好地避免过拟合和欠拟合，得到的结论也更有说服力。

5.3.2 模型与三种品牌的匹配程度与检验

（一）数据采样

数据附件一中的 1964 条数据中只有 99 位客户有购买意愿，类别的比例不均衡时，多数类的样本会被过多地关注，这样，少数类样本的分类性能就会受到影响，因此需

要对数据不平衡的问题进行处理，本文采用抽样 SMOTE 法解决标签类别不平衡问题，使得目标客户的购买意向 1 标签（购买）与 0 标签（不买）的比值调整为 0.6。

（二）训练集与验证集的划分

本文将把 1964 条数据集划分成训练集和验证集，训练集占比 0.8，验证集占比 0.2，进行 K 折交叉验证。

（三）模型训练与比较

使用 python 对五个模型在三个品牌的训练集上进行训练，对验证集进行预测，计算 F1 值和 AUC 值，计算结果如下：

表 4. 五种模型的 F1 与 AUC 值的比较结果

模型名称	品牌 1		品牌 2		品牌 3	
	F1 值	AUC 值	F1 值	AUC 值	F1 值	AUC 值
LR 模型	0.9041	0.9275	0.8950	0.9230	0.8684	0.9183
SVM 模型	0.9380	0.9487	0.9158	0.9236	0.9113	0.9058
随机森林模型	0.9590	0.9579	0.9598	0.9490	0.9181	0.9600
LightGBM 模型	0.9413	0.9550	0.9625	0.9599	0.9049	0.9511
LCV 模型	N.A.N.	0.9782	N.A.N.	0.9411	N.A.N.	0.9495

* N.A.N.即“not a number”，表示数据没有应用价值。

由表 3 中 F1、AUC 的结果可知，对品牌 1 和品牌 3 均适用于随机森林模型；品牌 2 则适合采用 LightGBM 模型。下图 7 为相应品牌与模型利用 python 软件绘制出的学习曲线。

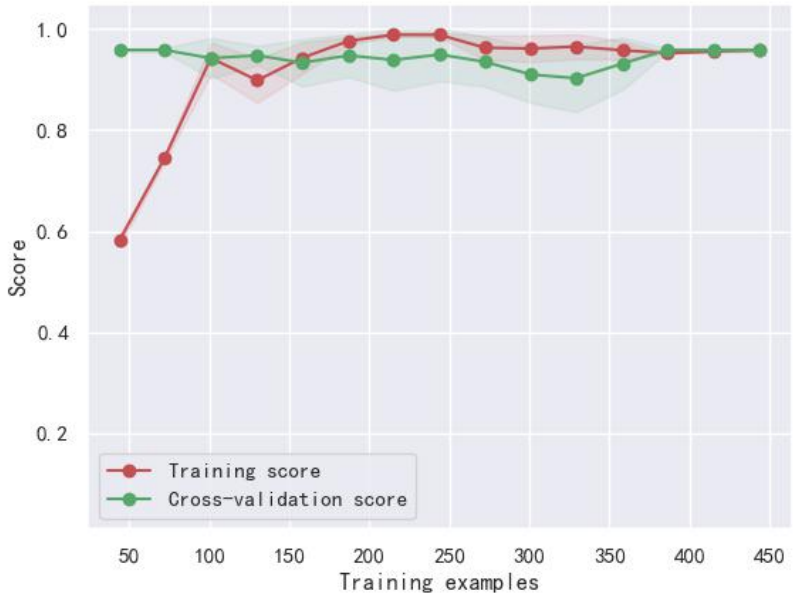

图 7. 品牌 1 随机森林学习曲线

表 5. 品牌 1 随机森林模型调参

参数名称	调参范围	调参结果	调参后 F1	调参后 AUC
决策树最大深度（max_depth）	1~100	5	0.9728	0.9756
决策树数目(n_estimates)	1~100	10		
分裂所需最少样本数(min_samples_split)	1~80	30		
叶节点最少承载样本数(min_samples_leaf)	1~50	3		

先通过网格搜索确定决策树数目和决策树最大深度，再确定分裂所需最少样本数和叶节点最少承载样本数，可以很直观地看到随机森林模型的表现性能更好，与比线性模型相比有着很大优势。并且在运行随机森林的时候发现代码的运行速度快，这是由于随机森林容易做成并行化方法，随机森林在训练时，树与树之间是相互独立的。

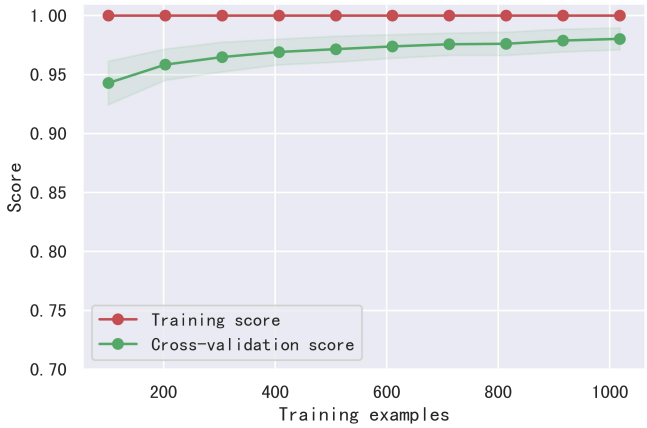

图 8. 品牌 2 LightGBM 学习曲线

表 6. 品牌 2 LightGBM 模型调参

参数名称	调参范围	调参结果	调参后 F1	调参后 AUC
决策树最大深度（max_depth）	1-25	7	0.9864	0.9710
最大叶子节点数（num_leaves）	5-100	35		
叶节点最小样本数(min_data_in_leaf)	1-40	20		
学习率（learning_rate）	0.001-0.1	0.015		
决策树数目（n_estimates）	1-100	8		

LightGBM 模型的参数较多，但是调参规则和每一个基于决策树的模型是差不多的，先确定 0.1 为初始学习率，可以使模型有更快的收敛速度，然后通过网格搜索确定决策树数目和决策树最大深度，再确定最大叶子节点数，最后调节叶节点最小样本数防止模型过拟合。

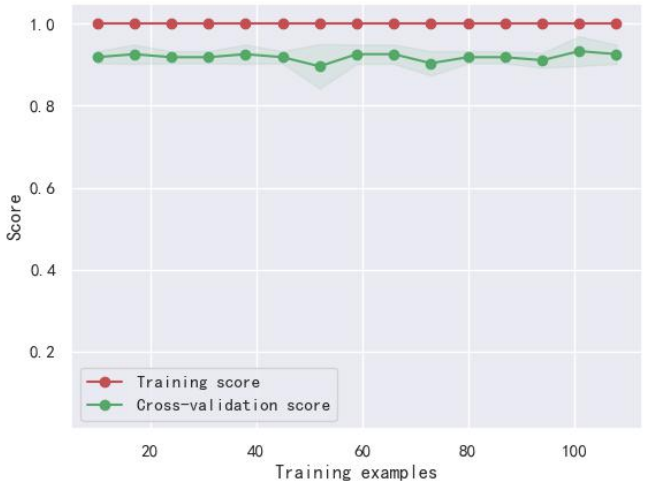

图 9. 品牌 3 随机森林学习曲线

表 7. 品牌 3 随机森林模型调参

参数名称	调参范围	调参结果	调参后 F1	调参后 AUC
决策树最大深度（max_depth）	1-100	8	0.9767	0.9896
决策树数目 (n_estimates)	1-100	10		
分裂所需最少样本数 (min_samples_split)	1-80	3		
叶节点最少承载样本数 (min_samples_leaf)	1-50	3		

由上述分析可知，品牌 1 和 3 与随机森林模型匹配性较好，品牌二与 LightGBM 模型的匹配性较好，故本文将利用随机森林、LightGBM 模型来预测附件三中 15 名目标客户购买电动汽车的可能性。

5.3.3 判断指定客户购买电动汽车的可能性

利用随机森林模型、LightGBM 模型、随机森林模型分别对品牌 1、2 和 3 的客户进行预测，结果如表 8 所示。

表 8. 附件三客户购买电动汽车的可能性

客户编号	品牌编号	购买可能性	预测
1	1	0.6	购买
2	1	0.08	不购买
3	1	0.03	不购买
4	1	0.05	不购买
5	1	0.28	不购买
6	2	0.99	购买
7	2	0.98	购买
8	2	0.01	不购买
9	2	0.01	不购买
10	2	0.01	不购买
11	3	0.1	不购买
12	3	0	不购买
13	3	0.2	不购买
14	3	0	不购买
15	3	0	不购买

由结果可知，附件三中 15 名待预测的目标客户中，仅有编号为 1、6、7 的客户有较大概率会选购品牌 1、品牌 2、品牌 2 的电动汽车，其他客户选购电动车的可能性较低。

5.4 问题四模型的建立与求解

本文从附件 3 中选取没有购买意愿，但购买意愿的预测值较高的客户，即编号为 3、9、14 号的目标客户，分别制定品牌 1、2、3 的销售策略。考虑到服务难度与提高的满意度百分点是成正比的，故本文的目标是使提高服务的难度最小的同时提高顾客满意度，从而提高购买概率。通过使用规划的方法，记录下满意度有提升的指标，结果如表 9~11 所示。

与此同时，本文基于问题二的结果，通过模型对目标客户的个人特征进行探究，通过聚焦目标客户的背景情况去实施销售策略，可以在不额外提高满意度的情况下提升客户的购车概率。

(一)品牌 1 的销售策略——以编号 3 的目标客户为例

基于问题二的结论，选择对品牌 1 销售影响较大的 a1、a2、a3、a4 进行讨论，再基于问题三的研究背景，选用随机森林作为模型进行预测。编号 3 的顾客 a1~a4 的初始满意度分数分别为 76.29、74.61、66.69、88.88 分，原始购买概率为 0.02。规划模型如式（17）~（21）所示。

购买概率：该销售部门的销售目标是在实施策略后能使目标客户购买电动车概率最大，即：

$$max\, Z_1 = \frac{\hat{y} - y}{y} \tag{1}$$

其中$\hat{y}$为提升服务后的购买意愿的预测概率，y 为原始预测概率。

提升的服务难度：考虑到提升服务满意度与服务难度成正比关系，故需控制提升服务的难度最小，即：

$$min\, Z_2 = \sum_{i=1}^{n} c_i \frac{\hat{x_i} - x_i}{x_i} \tag{2}$$

其中c_i为决策变量，c_i=1 为提升该服务，c_i=0 为不提升该服务。

提升的服务个数：经求解后发现，提升的服务个数越多，尽管保持总的满意度提升分数相同，但总的满意度都会超过 5%，不符合题意，故本文的目标是使控制提升的服务个数最少，即：

$$min\, Z_3 = \sum_{i=1}^{n} c_i \tag{3}$$

综上所述，销售部门要在服务难度的约束下使购买概率提升的百分比最大，并将服务个数降到最少，需将上述三个目标转化为单一目标[21,22]，列式可得：

$$min \sum_{i=1}^{n} c_i + \sum_{i=1}^{n} c_i \frac{\hat{x_i} - x_i}{x_i} - \frac{\hat{y} - y}{y} \tag{4}$$

控制参数的范围如下所示：

$$s.t \begin{cases} x_i \le \hat{x_i} \le 100 \\ c_i = 0 \text{ 或 } 1 (i = 1, \cdots, m) \\ \hat{y} \ge y \end{cases} \tag{5}$$

用 python 求解，得到以下结果：

表 9. 编号 3 的目标客户在求解后的结果

序号	a1/分	a2/分	a3/分	a4/分	购买概率
1	76.29	74.61	66.69	88.88	0.0255
2	76.29	74.61	86.69	88.88	0.0269
3	76.29	74.61	87.69	88.88	0.0514
4	76.29	74.61	88.69	88.88	0.0722
5	76.29	90.61	87.69	88.88	0.0803
6	76.29	90.61	88.69	88.88	0.1065
7	76.29	91.61	88.69	88.88	0.1092

由表 9.经计算后可知，序号为 4 的优化结果使 a3 提升了 22 分，服务满意度提升了 4.1%（<5%），且购买概率提升了 5%，满足题目要求。故针对编号 3 顾客的销售方向应着重考虑品牌 1（合资品牌）的经济性，也即耗能与保值率上。

因此对编号 3 顾客的销售方向应着重考虑品牌 1（合资品牌）的经济性，也即耗能与保值率上，在销售过程中，可以重点对该电动汽车的耗能进行全方位多角度的阐述，详细介绍电池容量，耗电率，使用年限等指标。

(二)品牌 2 的销售策略——以编号 9 的目标客户为例

基于问题二的结论，从目标客户的个人特征入手，通过 LightGBM 模型与 python 运行后我们发现，编号为 9 的目标客户个人年收入为 10 万（<25 万），相比同品牌其他客户有较高的购买概率，因此选择编号 9 作为销售对象，并在满意度上于下文进一步讨论。

从提高满意度的角度，选择对品牌 2 销售影响较大的 a1、a2、a3、a4、a5、a7 进行讨论，再基于问题三的研究背景，选用 LightBGM 作为模型进行预测。编号 3 的顾客 a1~a5、a7 的初始满意度分数分别为 82.41、88.92、85.17、85.61、85.63、82.35 分，原始购买概率为 0.01。

表 10. 编号 9 的目标客户在求解后的结果

序号	a1/分	a2/分	a3/分	a4/分	a5/分	a7/分	购买概率
1	85.41	88.92	85.17	85.61	85.63	82.35	0.0426
2	85.41	90.92	85.17	85.61	94.63	86.35	0.459
3	85.41	90.92	85.17	85.61	85.63	82.35	0.617
4	90.41	88.92	85.17	85.61	85.63	82.35	0.708
5	98.41	88.92	85.17	85.61	85.63	82.35	0.1084

由表 9.经计算后可知，序号 5 的优化结果使 a1 提升了 16 分，服务满意度提升了 2.43%（<5%），且购买概率提升了 9%，满足题目要求。

故针对编号 9 顾客的销售方向应着重考虑品牌 2（自主品牌）的电池技术性能，也即电池耐用和充电方便程度上，在销售过程中，销售人员可以详细阐述电池的容量，充电方式，充电桩地点分布，使目标客户对于该汽车的电池耐用度和充电便捷程度上，尽可能达到满意的水平。

(三)品牌 3 的销售策略——以编号 14 的目标客户为例

基于问题二的结论，从目标客户的个人特征入手，通过 LightGBM 模型与 python 运行后我们发现，编号为 14 的目标客户的全年房贷、车贷支出为 0，相比同品牌其他客户有较高的购买概率，因此选择编号 14 作为销售对象，并在满意度上于下文进一步讨论。

从提高满意度的角度，选择对品牌 3 销售影响较大的 a1、a2、a5 进行讨论，再基于问题三的研究背景，选用随机森林作为模型进行预测。编号 14 的顾客 a1、a2、a5 的初始满意度分数分别为 76.89、77.80、74.69 分，原始购买概率为 0.07。

表 11. 编号 14 的目标客户在求解后的结果

序号	a1/分	a2/分	a5/分	购买概率
1	76 .89	77 .80	74.69	0.0732
2	76 .89	77 .80	89.69	0.1084
3	76 .89	77 .80	90.69	0.2143
4	76 .89	77 .80	91.69	0.2237
5	76 .89	77 .80	92.69	0.2364
6	76 .89	93 .80	92.69	0.2382
7	76 .89	94 .80	90.69	0.2543
8	76 .89	94 .80	91.69	0.2637
9	76 .89	94 .80	92.69	0.2763
10	76 .89	95 .80	74.69	0.3360

由表 11.经计算后可知，序号为 10 的优化方案使 a2 提升了 17 分，服务满意度提升了 2.8%（<5%），且购买概率提升了 35%，满足题目要求。

故针对编号 14 顾客的销售方向应着重考虑品牌 3（新势力品牌）的舒适性，也即环保与空间座椅上，在销售过程中尽量营造良好的舒适体验感，辅以侧面手段，如播放令人放松的音乐等侧面方法，以打造轻松的环境。

（四）销售策略总结

综上所述，我们使用精准营销的策略，进行点对点的差异化营销。在没有购买意愿客户中，挑选经济条件较好的客户，对品牌一的客户应当重点对该电动汽车的耗能进行全方位多角度的阐述，对品牌二的客户详细阐述电池的容量，充电方式，充电桩地点分布，对品牌三的客户在销售过程中尽量营造良好的舒适体验感，辅以侧面手段，以打造轻松的环境。

5.5 问题五销售策略

本文基于问题一至四的结论，提出的模型具有很好的解释性，能考虑满意度与目标客户个人特征之间以及特征和模型之间的相关性，得到的结论更具说服力；且筛选出来的最终影响因素更具代表性。基于上述分析，本文提出以下销售策略：

(一)品牌分析

基于数据清洗工作的结果，对目标客户三个品牌的满意度进行比较分析。结果显示，目标客户比较青睐于合资品牌，因此可将销售的重心放在推广合资品牌的电动汽车上，自主、新势力品牌次之。

(二)人群洞察

在人群选择上，我们发现，户口在本城市、已婚、自由职业之或收入情况中等的家庭会更加倾向于购买电动汽车，为此可针对这些群体进行精准营销。三种品牌的销售都可从电池技术性能、舒适性入手为顾客进行介绍。此外，目标客户在选择合资品牌时还会关注经济性、安全性、驾驶操控性问题；经济性、安全性、动力性、外观内

饰是影响自主品牌销售的主要因素；新势力品牌可以电动车的动力性为重点。

(三)效果评估

本团队通过对五种模型进行比对、校验，发现随机森林适用于合资品牌与新势力品牌，自主品牌适合 LightGBM 模型，且两种模型的 F1 值、AUC 值均大于 97%，接近完美模型，因此可在这些模型的基础上进行精准营销。为了在提高顾客满意度与减少服务难度中找到优化方案，在建立模型的基础上得出结论：合资、自主和新势力品牌在其他条件不变的情况下，分别提升客户对经济性、电池性能和舒适性的满意度有助于提高客户的购车率。

综上所述，以上销售策略满足：需要在服务个数、服务难度尽可能小的情况下，提高客户满意度与购买率；对有一定特征的目标客户，如家庭住址在本城市、已婚、自由职业之或收入情况中等的家庭进行精准营销，从而助力销售部门实现效益的最大化。

六、 模型的评价与改进

6.1 模型的优点

1. 决策树模型如随机森林和 LightGBM 挖掘出来的分类规则准确性高，具有很好的解释性，便于理解。
2. 使用嵌入法进行特征选择，能够考虑特征之间的相关性，以及特征和模型之间的相关性，效果优于使用过滤方法的卡方检验，相关系数等方法，得到的结论也更有说服力。
3. 使用特征投票机制，考虑同一个指标在不同模型中出现的次数，筛选出来的最终影响因素可信度高，也更具有代表性。
4. 对于训练数据类别不平衡问题使用了 SMOTE 采样，一定程度上解决了目标客户购买人数过少的问题，减轻模型的欠拟合。
5. 设置 K 折交叉验证，通过 AUC 值和 F1 分数的平均值的结果，综合考量了各学习模型的预测准确度和稳定性，选出来的模型更具有代表性，泛化效果更好。

6.2 模型的缺点及改进方向

1. 使用机器学习模型提取主要特征虽然能加快训练速度，但也会使样本信息提取不完全导致部分信息丢失，后续可以多进行特征工程，尝试构造 Stacking 特征。
2. 整体样本数量偏少，在 LGBM 模型中在少量训练后快速收敛，可能会长出比较深的决策树，容易过拟合，模型泛化能力下降，后续可以尝试对该模型进行改进。
3. 随机森林算法可能有很多相似的决策树，掩盖了真实的结果，而且调参效率不高，无法控制模型内部的运行，只能在不同的参数和随机种子之间进行尝试，后续可以尝试加入更多模型进行比较分析。

七、 参考文献

[1] 王小峰,于志民.中国新能源汽车的发展现状及趋势[J].科技导报,34(17):13-18, 2016.

[2] 尹建杰. Logistic 回归模型分析综述及应用研究[D]. 黑龙江大学, 2011.[3] 孙烨. 两目标二分类 Logistic 回归模型的研究[D]. 北京工业大学.

[4] 郑峰. 基于二分类 Logistic 回归模型的企业网络营销统计分析[J]. 统计与决策, 2012, 000(023):196-198.

[5] 董纯洁. 基于实例与逻辑回归的多标签分类模型[D]. 南京大学, 2013.

[6] 姚燕云, 蔡尚真. 基于 LASSO 回归的葡萄酒评价研究[J]. 云南农业大学学报(自然科学), 2016, V31(002):294-302.

[7] 刘丽萍. 基于 Lasso 回归法的人口出生率影响因素分析[J]. 牡丹江师范学院学报：自然科学版(2):3.

[8] 谢玲, 刘琼荪. 集成 Logistic 与 SVM 的二分类算法[J]. 计算机工程与应用, 2011(29):149-150.

[9] 曲媛. 基于 SVM 的二分类不平衡数据问题研究[D]. 华南理工大学, 2009.

[10] 周涛丽. 基于支持向量机的多分类方法研究[D]. 电子科技大学.[11] 丁世飞, 齐丙娟, 谭红艳. 支持向量机理论与算法研究综述[J]. 电子科技大学学报, 2011, 40(001):2-10.

[12] 张建彬,霍佳震.基于 Stacking 模型融合的用户购买行为预测研究[J].上海管理科学,43(01):12-19, 2021.

[13] 郭萱. 基于随机森林的电影票房预测研究[D]. 中国石油大学(北京), 2018.

[14] 姚登举, 杨静, 詹晓娟. 基于随机森林的特征选择算法[J]. 吉林大学学报(工学版), 2014(01):142-146.

[15] 李欣海. 随机森林模型在分类与回归分析中的应用[J]. 应用昆虫学报(昆虫知识), 2013, 50(004):001190-1197.

[16] Wang B , Wu P , Chen Q , et al. Prediction and Analysis of Train Passenger Load Factor of High-Speed Railway Based on LightGBM Algorithm[J]. Journal of Advanced Transportation, 2021, 2021(4):1-10.

[17] Liang W , Luo S , Zhao G , et al. Predicting Hard Rock Pillar Stability Using GBDT, XGBoost, and LightGBM Algorithms[J]. 2020.

[18] 许姗姗.基于机器学习的商品销售预测的研究[J].统计与管理,2019(04):49-52.

[19] 左小德, 姚蓉静, 梁云. 优化方法在销售策略中的应用[J]. 暨南大学学报(自然科学与医学版), 2000.

[20] 田帅. 基于组合模型的销售预测应用研究[D]. 华南理工大学, 2016.

[21] 彭少明, 黄强, 刘涵,等. 黄河流域水资源可持续利用多目标规划模型研究[J]. 干旱区资源与环境, 2007.

[22] 林洪孝, 彭绪民. 城市水权分配机会的多目标规划模型[J]. 水利学报, 2005, 36(004):452-455.

附录

附录 1
介绍：支撑材料的文件列表
1. svm_get_features.py 使用 python 语言，用 SVM，LR，Lasso 模型提取原始数据特征 2. RF_get_features.py 使用 python 语言，用 RF(随机森林)回归模型提取原始数据特征 3. Lgb_get_features.py 使用 python 语言，用 LightGBM 模型提取原始数据特征 4. model_compare.py 输入附件三的数据，五折交叉验证，比较各个模型的优劣 5. forest_attribute.py 使用 python 语言，使用网络搜索调参，并画出学习曲线 6. random_forest_predict.py 使用 python 语言，使用随机森林和 lgbm 模型，附预测件三的数据 7. question4_forest_1.py 输入附件三的数据，遍历指标，解决问题 4 的品牌一的部分 8. question4_forset_2.py 输入附件三的数据，遍历指标，解决问题 4 的品牌三的部分 9. question4_lgbm_2.py 输入附件三的数据，遍历指标，解决问题 4 的品牌二的部分

附录 2
介绍：该代码是用 python 编写的，SVM，LR，Lasso 模型 提取特征
svm_get_features.py from sklearn.linear_model import LogisticRegression as LR from sklearn.svm import LinearSVC from sklearn.feature_selection import SelectFromModel from sklearn.linear_model import Lasso, LassoCV from sklearn.preprocessing import MinMaxScaler import pandas as pd import numpy as np df = pd.read_excel('2.xlsx')

```
# b13,14,15 处理
df.loc[(df['B13'] < df['B14']) | (df['B13'] < df['B15']), 'B13'] = np.nan
df.loc[df['B13'].isnull(), 'B13'] = df.loc[df['B13'].isnull(), 'B15'] * 1.667
df.loc[(df['B13'] < df['B14']) | (df['B13'] < df['B15']), 'B13'] = np.nan
df.loc[df['B13'].isnull(), 'B13'] = df.loc[df['B13'].isnull(), 'B14'] * 1.7142

x = df.iloc[:, 2:-1]
Y = df.iloc[:, -1]
Scaler = MinMaxScaler().fit(x)   #  标准化
X = Scaler.transform(x)
print(X.shape)
print(Y.shape)

# SVM 模型
lsvc = LinearSVC(C=3, penalty="l1", dual=False)   # 1-0.99; 2-3;3-0.35
sfm = SelectFromModel(lsvc, max_features=8).fit(X, Y)
print(sfm)
print(sfm.get_support())
print(df.columns[2:-1][sfm.get_support()])

# lasso 回归模型
clf = LassoCV()
sfm = SelectFromModel(clf, max_features=8).fit(X, Y)
print(sfm)
print(sfm.get_support())
print(df.columns[2:-1][sfm.get_support()])

# LR 逻辑回归模型
sfm = SelectFromModel(estimator=LR(C=7), max_features=8).fit(X, Y)
print(sfm)
print(sfm.get_support())
print(df.columns[2:-1][sfm.get_support()])
```

附录 3
介绍：该代码是用 python 编写的，随机森林提取特征

RF_get_features.py

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display
sns.set(style="darkgrid",palette='deep')
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
%matplotlib inline
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

df = pd.read_excel('2_del.xlsx')
df1 = df[df.品牌类型  == 1]
df2 = df[df.品牌类型  == 2]
df3 = df[df.品牌类型  == 3]
#随机森林

X = df3.iloc[:,2:27]
# X.drop(['B1','B6','B7'],axis=1)
# X.sample(n=100,random_state=0)
target = df3.iloc[:,27]

x_train, x_test, y_train, y_test = train_test_split(X, target, test_size = 0.2, random_state =
0)
feat_labels = df.columns[2:27]
forest = RandomForestClassifier(n_estimators=10000, random_state=0, n_jobs=-1)
forest.fit(x_train, y_train)

#  特征重要性
l = []
importances = forest.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(x_train.shape[1]):
```

```
        print("%2d) %-*s %f" % (f + 1, 30, feat_labels[indices[f]],
importances[indices[f]]))
        l.append((feat_labels[indices[f]],importances[indices[f]]))
l=pd.DataFrame(l)
```

附录 6

介绍：该代码是用 python 编写的，LightGBM 提取特征

Lgb_get_features.py

```
from sklearn.metrics import mean_squared_error
import lightgbm as lgb
from sklearn.model_selection import train_test_split

df = pd.read_excel('2_del.xlsx')
X = df.iloc[:,0:12]
# X.drop(['B1','B6','B7'],axis=1)
# X.sample(n=100,random_state=0)
target = df.iloc[:,12]
X_train, X_test, y_train, y_test = train_test_split(X, target,
test_size=0.2,random_state=0)

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score,roc_auc_score,f1_score
evals_result = {}

valid_sets = [X_train, X_test]
valid_name = ['train', 'eval']

# 创建成 lgb 特征的数据集格式
lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

# 将参数写成字典下形式
params = {
```

```
    'task': 'train',
    'boosting_type': 'gbdt',  # 设置提升类型
    'min_data_in_leaf': 10,
     'max_depth': -1,
    'objective': 'binary',  # 目标函数
    'metric': {'binary_logloss', 'auc','f1'},  # 评估函数
    'num_leaves':3,  # 叶子节点数
    'learning_rate': 0.0001,  # 学习速率
    'feature_fraction': 0.8,  # 建树的特征选择比例
    'bagging_fraction': 0.8,  # 建树的样本采样比例
    "lambda_l1": 0.1,
    'verbose': -1  ,# <0 显示致命的, =0 显示错误 (警告), >0 显示信息
    "nthread": -1,
     'n_estimators' : 500,
    'is_unbalance':True,
}

# 训练 cv and train
gbm = lgb.train(params, lgb_train,
                        valid_names=valid_name,
                        num_boost_round=500,
                        valid_sets=lgb_eval,
                        evals_result=evals_result,
                        early_stopping_rounds=20)

# 预测数据集
y_pred = gbm.predict(X_test, num_iteration=gbm.best_iteration)
lgb.plot_metric(evals_result, metric='auc')

plt.figure(figsize=(12,6))
lgb.plot_importance(gbm, max_num_features=30)
plt.title("Featurertances")
# plt.savefig(f'F:./图片/特征重要度 1.png',dpi=600)

importance = gbm.feature_importance(importance_type='split')
feature_name = gbm.feature_name()
# for (feature_name,importance) in zip(feature_name,importance):
#         print (feature_name,importance)
```

```
feature_importance =
pd.DataFrame({'feature_name':feature_name,'importance':importance} )
feature_importance.sort_values(['importance'],ascending=False)

lgb_predictors = [i for i in X_train.columns[2:27]]
pd.Series(gbm.feature_importance(), lgb_predictors).sort_values(ascending=False)
```

附录 4

介绍：该代码是用 python 编写的，比较模型

model_compare.py

```
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import classification_report
import pandas as pd
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import roc_auc_score

df = pd.read_excel('2_del.xlsx')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print(Counter(y))
# 定义 SMOTE 模型，random_state 相当于随机数种子的作用
smo = SMOTE(sampling_strategy=0.58, random_state=2021)
X_smo, y_smo = smo.fit_resample(X, y)
print(Counter(y_smo))

Scaler = MinMaxScaler().fit(X_smo)# 数据标准化
X_trainStd = Scaler.transform(X_smo)

import pandas as pd
import warnings
from sklearn.preprocessing import scale
```

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import lightgbm as lgbm
from sklearn.linear_model import LassoCV

# 划分为 5 折交叉验证数据集
df_y = y_smo
df_X = X_trainStd
# 构建模型
lr = LogisticRegression(random_state=2021, tol=1e-6)  # 逻辑回归模型
svm = SVC(probability=True, random_state=2021, tol=1e-6)  # SVM 模型
forest = RandomForestClassifier(n_estimators=100, random_state=2021)  # 随机森林
gbm = lgbm.LGBMClassifier(random_state=2021)  # lgb
LCV = LassoCV(random_state=2021)

def muti_score(model):
    warnings.filterwarnings('ignore')
    f1_score = cross_val_score(model, df_X, df_y, scoring='f1', cv=5)
    auc = cross_val_score(model, df_X, df_y, scoring='roc_auc', cv=5)
    print("F1_score:", f1_score.mean())
    print("AUC:", auc.mean())

model_name = ["lr",  "forest", "svm",  "gbm", "LCV"]  # , "
for name in model_name:
    model = eval(name)
    print(name)
    muti_score(model)
```

附录 5

介绍：该代码是用 python 编写的，绘制学习曲线

forest_attribute.py

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
```

```
import pandas as pd
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import roc_auc_score, f1_score
from sklearn.model_selection import GridSearchCV
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display

sns.set(style="darkgrid",palette='deep')
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False

df = pd.read_excel('3_del.xlsx')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print(Counter(y))
# 定义 SMOTE 模型，random_state 相当于随机数种子的作用
smo = SMOTE(sampling_strategy=0.6, random_state=2021)
X_smo, y_smo = smo.fit_resample(X, y)
print(Counter(y_smo))
## 将数据划分为训练集测试集
X_train, X_test, \
y_train, y_test = \
      train_test_split(X_smo, y_smo,
                               test_size=0.2, random_state=2021)

Scaler = MinMaxScaler().fit(X_train)## 数据标准化
X_trainStd = Scaler.transform(X_train)
X_testStd = Scaler.transform(X_test)

rf = RandomForestClassifier(n_estimators=100,
random_state=2021,max_depth=15,min_samples_split=2,min_samples_leaf=1)
rf.fit(X_trainStd, y_train)
pred = rf.predict(X_testStd)
print('f1:', f1_score(y_test, pred))
```

```
print('auc:', roc_auc_score(y_test, pred))


# 2.我们首先对 n_estimators 进行网格搜索：
param_test1 = {'n_estimators': np.arange(30, 400)}
gsearch1 = GridSearchCV(estimator=RandomForestClassifier(min_samples_split=100,

min_samples_leaf=20, max_depth=8, max_features='sqrt',

random_state=2021),
                        param_grid=param_test1, scoring='roc_auc', cv=5)
gsearch1.fit(X_trainStd, y_train)
print('best_estimator:', gsearch1.best_params_, gsearch1.best_score_)

# 3.接着我们对决策树最大深度 max_depth 和内部节点再划分所需最小样本数
min_samples_split 进行网格搜索。
param_test2 = {'max_depth': np.arange(1, 100), 'min_samples_split': np.arange(1, 80)}
gsearch2 = GridSearchCV(estimator=RandomForestClassifier(n_estimators=219,

min_samples_leaf=20, max_features='sqrt', oob_score=True,

random_state=2021),
                        param_grid=param_test2, scoring='roc_auc', cv=5)
gsearch2.fit(X_trainStd, y_train)
print(gsearch2.best_params_, gsearch2.best_score_)

param_test3 = {'min_samples_leaf': np.arange(1, 50)}
gsearch3 = GridSearchCV(estimator=RandomForestClassifier(n_estimators=219,
max_depth=4,

min_samples_leaf=20, max_features='sqrt', oob_score=True,

random_state=2021),
                        param_grid=param_test3, scoring='roc_auc', cv=5)
gsearch3.fit(X_trainStd, y_train)
print(gsearch3.best_params_, gsearch3.best_score_)

from sklearn.svm import LinearSVC
```

```
from sklearn.model_selection import learning_curve

# 绘制学习曲线，以确定模型的状况
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure()
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=5, n_jobs=4, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r",
             label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g",
             label="Cross-validation score")
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    plt.legend(loc="best")
    plt.grid("on")
    if ylim:
        plt.ylim(ylim)
    plt.title(title)
    plt.show()

# 放在 main 里，使用多线程
if __name__ == "__main__":
    plot_learning_curve(rf, "",
                        X, y, ylim=(0.01, 1.05),
                        train_sizes=np.linspace(.1, 1, 15))
```

附录 6

介绍：该代码是用 python 编写的，预测附件三结果

random_forest_predict.py

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import roc_auc_score, f1_score
from sklearn.model_selection import GridSearchCV
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display

sns.set(style="darkgrid",palette='deep')
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False

df = pd.read_excel('1_del.xlsx')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print(Counter(y))
# 定义 SMOTE 模型，random_state 相当于随机数种子的作用
smo = SMOTE(sampling_strategy=0.6, random_state=2021)
X_smo, y_smo = smo.fit_resample(X, y)
print(Counter(y_smo))
## 将数据划分为训练集测试集
X_train, X_test, \
y_train, y_test = \
      train_test_split(X_smo, y_smo,
                                 test_size=0.2, random_state=2021)

Scaler = MinMaxScaler().fit(X_train)     ## 数据标准化
```

```
X_trainStd = Scaler.transform(X_train)
X_testStd = Scaler.transform(X_test)

# 随机森林预测
rf = RandomForestClassifier(n_estimators=100, max_depth=3, min_samples_leaf=3,
min_samples_split=54, random_state=2021)
rf.fit(X_trainStd, y_train)
pred = rf.predict(X_testStd)
print('f1:', f1_score(y_test, pred))
print('auc:', roc_auc_score(y_test, pred))

df1 = pd.read_excel('test_1.xlsx')
df1 = df1.iloc[:, :8]
Scaler = MinMaxScaler().fit(df1)
df2 = Scaler.transform(df1)
print(df2)
pred_1 = rf.predict(df2)
pred2 = rf.predict_proba(df2)
print(pred_1)
print(pred2)

# lgbm 预测
gbm = lgbm.LGBMClassifier(max_depth=7,random_state=2021,num_leaves=35)
gbm.fit(X_trainStd, y_train)
pred = gbm.predict(X_testStd)
print('f1:', f1_score(y_test, pred))
print('auc:', roc_auc_score(y_test, pred))
df1 = pd.read_excel('test_2.xlsx')
df1 = df1.iloc[:, :-3]
print(df1)
Scaler = MinMaxScaler().fit(df1)
df2 = Scaler.transform(df1)
pred_1 = gbm.predict(df2)
pred2 = gbm.predict_proba(df2)
print(pred_1)
print(pred2)
```

附录 7

介绍：该代码是用 python 编写的，解决问题 4 的品牌一的部分

question4_forest_1

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import roc_auc_score, f1_score
from sklearn.model_selection import GridSearchCV
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display
import lightgbm as lgbm

sns.set(style="darkgrid",palette='deep')
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False

df = pd.read_excel('1_del.xlsx')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print(Counter(y))
# 定义 SMOTE 模型，random_state 相当于随机数种子的作用
smo = SMOTE(sampling_strategy=0.6, random_state=2021)
X_smo, y_smo = smo.fit_resample(X, y)
print(Counter(y_smo))
## 将数据划分为训练集测试集
X_train, X_test, \
y_train, y_test = \
      train_test_split(X_smo, y_smo,
                                 test_size=0.2, random_state=2021)
```

```
Scaler = MinMaxScaler().fit(X_train)
X_trainStd = Scaler.transform(X_train)   ## 数据标准化
X_testStd = Scaler.transform(X_test)

rf = RandomForestClassifier(n_estimators=100, max_depth=3, min_samples_leaf=3,
min_samples_split=54, random_state=2021)
rf.fit(X_trainStd, y_train)
pred = rf.predict(X_testStd)
# print(pred)
print('f1:', f1_score(y_test, pred))
print('auc:', roc_auc_score(y_test, pred))

df1 = pd.read_excel('test_1.xlsx')
df1 = df1.iloc[:, 0:8]
num = 4
print(df1.iloc[num])
min0 = 1
max1 = 0
for i1 in range(76, 100):
    for i2 in range(74, 100):
        for i3 in range(66, 100):
            for i4 in range(88, 100):
                df1.iloc[num, 0] = i1
                df1.iloc[num, 1] = i2
                df1.iloc[num, 2] = i3
                df1.iloc[num, 3] = i4
                Scaler = MinMaxScaler().fit(df1)
                df2 = Scaler.transform(df1)

                l1 = []
                l1.append(np.array(df2[num]).tolist())
                pred_1 = rf.predict(l1)
                pred2 = rf.predict_proba(l1)
                if min0 > pred2[0][0]:
                    min0 = pred2[0][0]
                    print('min', min0)
                    i1_0 = i1
                    i2_0 = i2
```

```
                    i3_0 = i3
                    i4_0 = i4
                    print('各个 i：',i1_0, i2_0, i3_0, i4_0)
                if max1 < pred2[0][1]:
                    max1 = pred2[0][1]
                    print('max', max1)
```

附录 8

介绍：该代码是用 python 编写的，解决问题 4 的品牌三的部分

question4_forest_3.py

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import roc_auc_score, f1_score
from sklearn.model_selection import GridSearchCV
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display
import lightgbm as lgbm

sns.set(style="darkgrid",palette='deep')
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False

df = pd.read_excel('3_del.xlsx')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print(Counter(y))
# 定义 SMOTE 模型，random_state 相当于随机数种子的作用
smo = SMOTE(sampling_strategy=0.6, random_state=2021)
X_smo, y_smo = smo.fit_resample(X, y)
```

```
print(Counter(y_smo))
X_train, X_test, \
y_train, y_test = \          ## 将数据划分为训练集测试集
      train_test_split(X_smo, y_smo,
                                test_size=0.2, random_state=2021)

Scaler = MinMaxScaler().fit(X_train)   ## 数据标准化
X_trainStd = Scaler.transform(X_train)
X_testStd = Scaler.transform(X_test)

rf = RandomForestClassifier(n_estimators=100, max_depth=3, min_samples_leaf=3,
min_samples_split=54, random_state=2021)
rf.fit(X_trainStd, y_train)
pred = rf.predict(X_testStd)
# print(pred)
print('f1:', f1_score(y_test, pred))
print('auc:', roc_auc_score(y_test, pred))

df1 = pd.read_excel('test_3.xlsx')
df1 = df1.iloc[:, 0:7]
num = 4
print(df1.iloc[num])
min0 = 1
max1 = 0
for i1 in range(76, 100):
      for i2 in range(77, 100):
            for i3 in range(74, 100):
                  df1.iloc[num, 0] = i1
                  df1.iloc[num, 1] = i2
                  df1.iloc[num, 2] = i3
                  Scaler = MinMaxScaler().fit(df1)
                  df2 = Scaler.transform(df1)

                  l1 = []
                  l1.append(np.array(df2[num]).tolist())
                  pred_1 = rf.predict(l1)
                  pred2 = rf.predict_proba(l1)
                  if min0 > pred2[0][0]:
```

```
                    min0 = pred2[0][0]
                    print('min', min0)
                    i1_0 = i1
                    i2_0 = i2
                    i3_0 = i3
                    print('各个 i：',i1_0, i2_0, i3_0)
                if max1 < pred2[0][1]:
                    max1 = pred2[0][1]
                    print('max', max1)
```

附录 9

介绍：该代码是用 python 编写的，解决问题 4 的品牌二的部分

question4_lgbm_2.py

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.metrics import roc_auc_score, f1_score
from sklearn.model_selection import GridSearchCV
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from IPython.display import display
import lightgbm as lgbm

sns.set(style="darkgrid",palette='deep')
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False

df = pd.read_excel('2_del.xlsx')
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
print(Counter(y))
# 定义 SMOTE 模型，random_state 相当于随机数种子的作用
```

```
smo = SMOTE(sampling_strategy=0.6, random_state=2021)
X_smo, y_smo = smo.fit_resample(X, y)
print(Counter(y_smo))
## 将数据划分为训练集测试集
X_train, X_test, \
y_train, y_test = \
      train_test_split(X_smo, y_smo,
                                 test_size=0.2, random_state=2021)

Scaler = MinMaxScaler().fit(X_train)   ## 数据标准化
X_trainStd = Scaler.transform(X_train)
X_testStd = Scaler.transform(X_test)
gbm = lgbm.LGBMClassifier(random_state=2021)   # lgb
gbm.fit(X_trainStd, y_train)
pred = gbm.predict(X_testStd)
# print(pred)
print('f1:', f1_score(y_test, pred))
print('auc:', roc_auc_score(y_test, pred))

df1 = pd.read_excel('test_2.xlsx')
df1 = df1.iloc[:, 0:12]
num = 3
print(df1.iloc[num])
min0 = 1
max1 = 0
for i1 in range(82, 100):
      for i2 in range(88, 100):
            for i3 in range(85, 100):
                  for i4 in range(86, 100):
                        for i5 in range(86, 100):
                              for i6 in range(82, 100):
                                    df1.iloc[num, 0] = i1
                                    df1.iloc[num, 1] = i2
                                    df1.iloc[num, 2] = i3
                                    df1.iloc[num, 3] = i4
                                    df1.iloc[num, 4] = i5
                                    df1.iloc[num, 5] = i6
                                    Scaler = MinMaxScaler().fit(df1)
```

```
                df2 = Scaler.transform(df1)
                l1 = []
                l1.append(np.array(df2[num]).tolist())
                pred_1 = gbm.predict(l1)
                pred2 = gbm.predict_proba(l1)
                if min0 > pred2[0][0]:
                    min0 = pred2[0][0]
                    print('min', min0)
                    i1_0 = i1
                    i2_0 = i2
                    i3_0 = i3
                    i4_0 = i4
                    i5_0 = i5
                    i6_0 = i6
                    print('各个 i：', i1_0, i2_0, i3_0, i4_0, i5_0, i6_0)
                if max1 < pred2[0][1]:
                    max1 = pred2[0][1]
                    print('max', max1)
```